React Components

React, a JavaScript library, is the most popular front-end framework in tech by usage. Components are one of React's fundamental building blocks. Here's what to know about them.

React components:

- *Serve the same purpose as JavaScript functions, except they effectively divide the UI into reusable components that return HTML.*
- *Are independent mixtures of JavaScript and HTML.*
- *Come in two types: class and functional components.*
- *Class components took a back seat to functional components with React version 16.8 but are still supported.*

Example:

```
function WelcomeMessage() {
  return (
    <p>Hello, World</p>
  )
}
export default WelcomeMessage;
```

This code defines a function named WelcomeMessage() that can be rendered in place of a placeholder <WelcomeMessage /> in the main JavaScript file. You just need to import the function into the main JS file:

```
import WelcomeMessage from "./WelcomeMessageComponent";

return (
<WelcomeMessage />
)
```

Class vs functional components:

Class components:

- *Extend from React.Component*
- *Known as stateful components*
- *Respond to lifecycle events*
- *Maintain state information*
- *Support props*
- *Require a constructor to store state before they can be used to pass props to the parent class*
- *Require a render function that returns an HTML element*

Functional components:

- *Don't extend from React.Component*
- *Known as stateless components*
- *Don't respond to lifecycle events*
- *Don't maintain state information*
- *Accept any type of data (props)*
- *Don't support a constructor*
- *Return HTML elements or nothing*
- *Support React 16.8 hooks*

Key terms:

Constructor: *The method that initializes a React object's state. It's automatically called when an object is created in a class and before a component is mounted.*

Hooks: *Allow you to "hook" into React features such as state and lifecycle methods in functional components.*

Lifecycle events: *Each component has a lifecycle that you can monitor and manipulate during its four main phases: mounting, updating, unmounting, and error handling.*

Lifecycle methods: *Methods you can invoke during lifecycle phases to update the DOM to reflect new state information.*

Props: *Props (properties) are passed to React components. Akin to function arguments in JavaScript and attributes in HTML.*

React.Component: *A class in React extended to create class components.*

Render function: *A required method in class components that returns an HTML element.*

State: *An updatable structure that holds component data and makes a component dynamic and interactive.*

Class component example:

```
import { Component } from 'react';
class MyComponent extends React.Component {
  constructor(props) {
    super(props);
    this.state = { currState: true }
  }
  render() {
    <div>
      <p>Hello, World!</p>
    </div>
  }
}
```

Functional component examples:

Using function keyword

```
function MyComponent(props) {
  return (
    <div>
      <p>Hello, World</p>
      <p>Have a nice day!</p>
    </div>
  );
}
```

Using arrow function syntax

```
const MyComponent = (props) => {
  return (
    <div>
      <p>Hello, World</p>
      <p>Have a nice day!</p>
    </div>
  );
}
```

educative